



1/20

FIG. 1
PRIOR ART

1 c:\collections
2 notes.txt
3 myletter.doc
4 c-myhomepage
5
6 s
7 homepage.html
8 myphoto.jpg

FIG. 2

1 c:\collections
2 notes.txt
3 myletter.doc
4 c-myhomepage
5 collspec
6 s
7 homepage.html
8 myphoto.jpg

100

FIG. 3

1 collection c-myhomepage
2 coll-type cf-web-page
3 coll-home cf-colls:mysite.com:c-myhomepage
4 coll-desc A sample homepage collection
5 end-collection

102





2/20

FIG. 4

```
1  c:\collections
2    programs
3      helloworld
4        c-hello-library
5        c-hello

6    c-myprogram

7    parts
8      c-include-files
9      c-library-one
10     c-library-two

11   webstuff
12     c-myhomepage
```

FIG. 5

```
1  # Pathnames showing filesystem location of collections
2
3  c:\collections\programs\helloworld\c-hello-library
4  c:\collections\programs\helloworld\c-hello
5  c:\collections\c-myprogram
6  c:\collections\parts\c-include-files
7  c:\collections\parts\c-library-one
8  c:\collections\parts\c-library-two
9  c:\collections\webstuff\c-myhomepage
```



FIG. 6
PRIOR ART

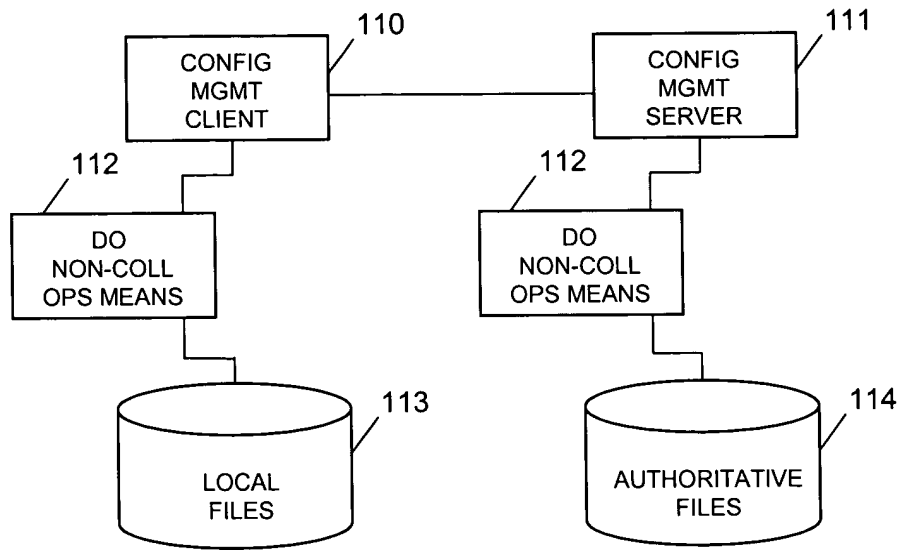
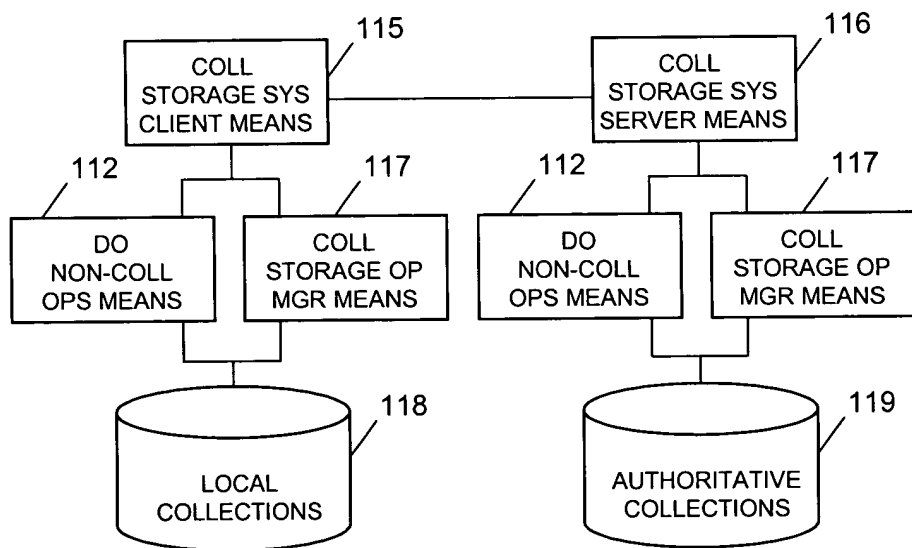


FIG. 7





4/20

FIG. 8

```
1  # Five components of a full collection reference
2  #
3  # <coll-name><scope arguments><content selector names>
4  # where <coll-name> = category:authority:collection
5  #
6  category    the hierarchical category name of the coll
7              e.g. site/prod/release4
8  authority   the authority name responsible for managing the coll
9              e.g. colls.mysite.com
10 collection  the collection name
11             e.g. mycoll
12 scope-args  command line arguments that select scope within coll
13             e.g. -recursive -new -changed -locked
14 selectors   names of categories, directories, or files
15             e.g. mydir, mydir/myfile.c, collspec.txt
```

FIG. 9

```
1  # Example collection reference for collection of FIG 2
2
3  # coll c-myhomepage in cat cf-colls at mysite.com
4  cf-colls:mysite.com:c-myhomepage
```



FIG. 10

1	# Shortcut collection references and their meanings	
2	#	
3	# Shortcut	Meaning
4		
5	cat:auth:coll	a full collection name reference
6	cat:auth:	all collections in category at authority
7	cat::coll	this coll in this cat at default authority
8	cat::	all colls in this cat at default authority
9	:auth:coll	this coll in all cats at this authority
10	:auth:	all colls in all cats at this authority
11		
12	::	current coll if inside a coll; invalid outside a coll
13	::.	current coll and current dir if inside; invalid outside
14	::mydir	current coll and mydir if inside; invalid outside



6/20

FIG. 11

```
1  # Structure of a collection symbolic job request
2  #
3  # "do this (task) to that (collection reference)"
4  # <task-name> <collection-ref>
5  # where <collection-ref> = category:authority:collection
6  #
7  category    the hierarchical category name of the coll
8              e.g. site/prod/release4
9  authority    the authority name responsible for managing the coll
10             e.g. colls.mysite.com
11  collection  the collection name
12             e.g. mycoll
```

FIG. 12

```
1  # Example collection symbolic job requests
2  # (using collections from FIGs 2-3)
3
4  # rebuild coll c-myhomepage in cat cf-colls at mysite.com
5  rebuild cf-colls:mysite.com:c-myhomepage
6
7  # rebuild all collections in category cf-colls at mysite.com
8  rebuild cf-colls:mysite.com:
```





7/20

FIG. 13

```
1 # Expanded collection list for a symbolic job request
2 #
3 # <task-name> <collection-ref>
4 # rebuild cf-colls:mysite.com:
5 #
6 cf-colls:mysite.com:c-hello-library
7 cf-colls:mysite.com:c-hello
8 cf-colls:mysite.com:c-myprogram
9 cf-colls:mysite.com:c-include-files
10 cf-colls:mysite.com:c-library-one
11 cf-colls:mysite.com:c-library-two
12 cf-colls:mysite.com:c-myhomepage
```

FIG. 14

```
1 # Expanded (sorted) visit order list for a symbolic job request
2 #
3 # <task-name> <collection-ref>
4 # rebuild cf-colls:mysite.com:
5 #
6 cf-colls:mysite.com:c-include-files      10
7 cf-colls:mysite.com:c-library-two       49
8 cf-colls:mysite.com:c-hello-library     50
9 cf-colls:mysite.com:c-library-one       50
10 cf-colls:mysite.com:c-hello            100
11 cf-colls:mysite.com:c-myhomepage        100
12 cf-colls:mysite.com:c-myprogram        100
```





8/20

FIG. 15

```
1 # Expanded platform list for cf-colls:mysite.com:c-myprogram
2 # Platform names are user-defined, and are not trademarks
3 win2000
4 win98
5 win95
6 linux2
```

FIG. 16

```
1 # Expanded platform list for cf-colls:mysite.com:c-myhomepage
2 #
3 win2000
```

FIG. 17

```
1 # Expanded job triplet list for a single collection
2 #
3 cf-colls:mysite.com:c-myprogram    100    win2000
4 cf-colls:mysite.com:c-myprogram    100    win98
5 cf-colls:mysite.com:c-myprogram    100    win95
6 cf-colls:mysite.com:c-myprogram    100    linux2
```

FIG. 18

```
1 # Expanded job triplet list for a single collection
2 #
3 cf-colls:mysite.com:c-myhomepage 100    win2000
```





9/20

FIG. 19

```
1  # Expanded triplet list for a symbolic job request
2  #
3  # rebuild cf-colls:mysite.com:

4  cf-colls:mysite.com:c-include-files      10    win2000
5  cf-colls:mysite.com:c-include-files      10    win98
6  cf-colls:mysite.com:c-include-files      10    win95
7  cf-colls:mysite.com:c-include-files      10    linux2

8  cf-colls:mysite.com:c-library-two        49    win2000
9  cf-colls:mysite.com:c-library-two        49    win98
10 cf-colls:mysite.com:c-library-two        49    win95
11 cf-colls:mysite.com:c-library-two        49    linux2

12 cf-colls:mysite.com:c-hello-library      50    win2000
13 cf-colls:mysite.com:c-hello-library      50    win98
14 ...                                     ...    ...
15
16 cf-colls:mysite.com:c-library-one        50    win2000
17 cf-colls:mysite.com:c-library-one        50    win98
18 ...                                     ...    ...

19 cf-colls:mysite.com:c-hello             100    win2000
20 cf-colls:mysite.com:c-hello             100    win98
21 ...                                     ...    ...

22 cf-colls:mysite.com:c-myhomepage        100    win2000

23 cf-colls:mysite.com:c-myprogram         100    win2000
24 cf-colls:mysite.com:c-myprogram         100    win98
25 cf-colls:mysite.com:c-myprogram         100    win95
26 cf-colls:mysite.com:c-myprogram         100    linux2
```



10/20

FIG. 20

```
1  /* data structure for holding expanded job info */
2  csje-info {
3      + original symbolic task name
4      + original collection-reference
5      + expanded-coll-list
6          + coll-structure-1
7              + coll-name
8              + visit-order
9              + platform-list
10             + platform-1
11             + platform-2
12             + platform-...
13             + other-coll-info
14          + coll-structure-2
15              + coll-name
16              + visit-order
17              + platform-list ...
18              ...
19          + coll-structure-3
20              ...
21      + other expansion info
22  }
```

11/20

FIG. 21

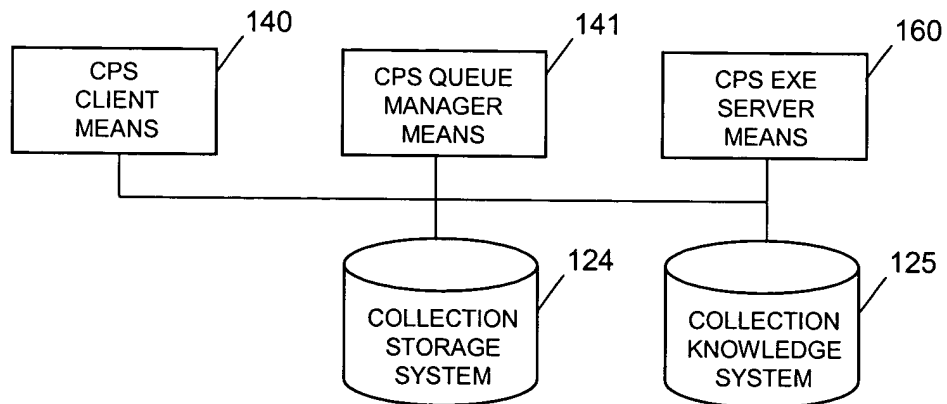


FIG. 22

- 1 # Simplified pseudocode algorithm for a CPS system
- 2 #
- 3 CPS Client receives symbolic job request, passes it to Queue Manager
- 4 CPS Queue Manager expands symbolic request into detailed job requests
- 5 CPS Queue Manager dispatches detailed requests to exe servers
- 6 CPS Execution Servers perform requested computations
- 7 CPS Queue Manager reports job results to request originator
- 8 All modules may make use of CSS 124 and CKS 125 systems

12/20

FIG. 23

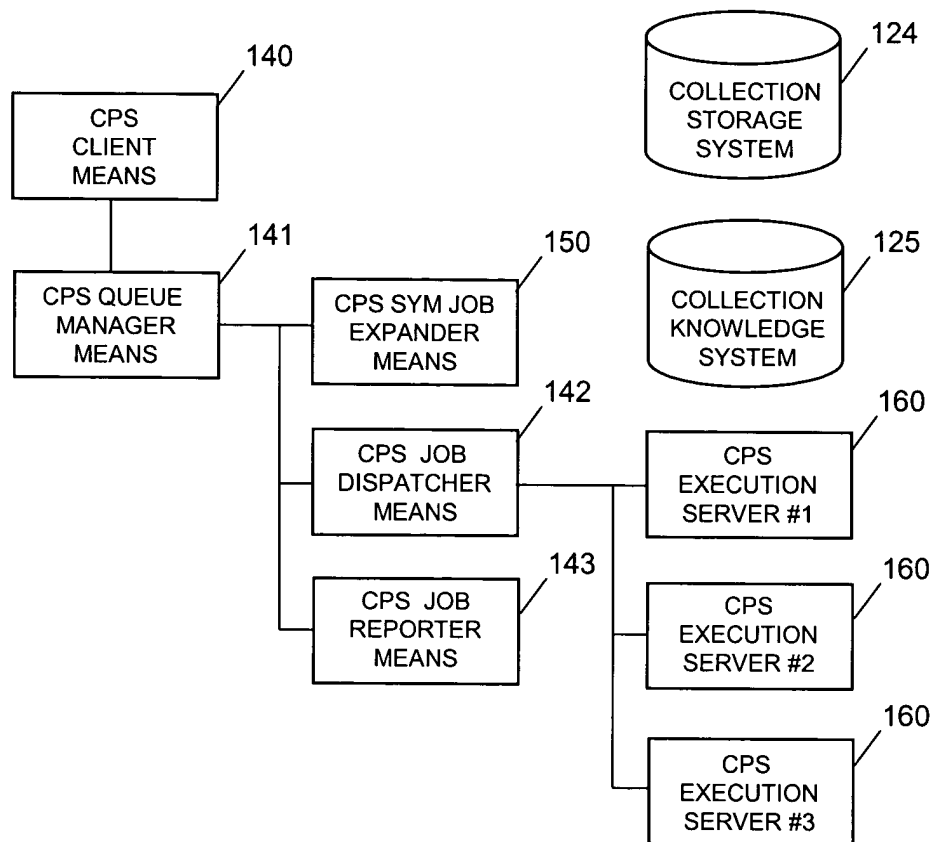


FIG. 24

- 1 # Simplified pseudocode algorithm for a collection processing system
- 2 CPS client receives job request, passes it to Queue Manager
- 3 Queue mgr calls internal job expander to expand job request
- 4 Queue mgr calls job dispatcher to expand and dispatch job triplets
- 5 Job dispatcher distributes second-level cmds among N exe servers
- 6 Job reporter means aggregates and reports job results
- 7 All modules may use CSS 124 and CKS 125, as required

FIG. 25

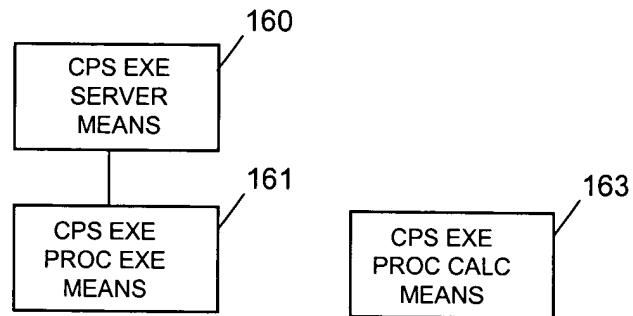


FIG. 26

- 1 # Simplified pseudocode algorithm for a CPS execution server
- 2 #
- 3 Receive 2nd level prologue/main/epilogue cmd from job dispatcher
- 4 Expand 2nd level command into 3rd level command
- 5 Call CPS Exe Process Execution Means to execute 3rd level cmd
- 6 Report job results back to CPS Job Dispatcher

FIG. 27

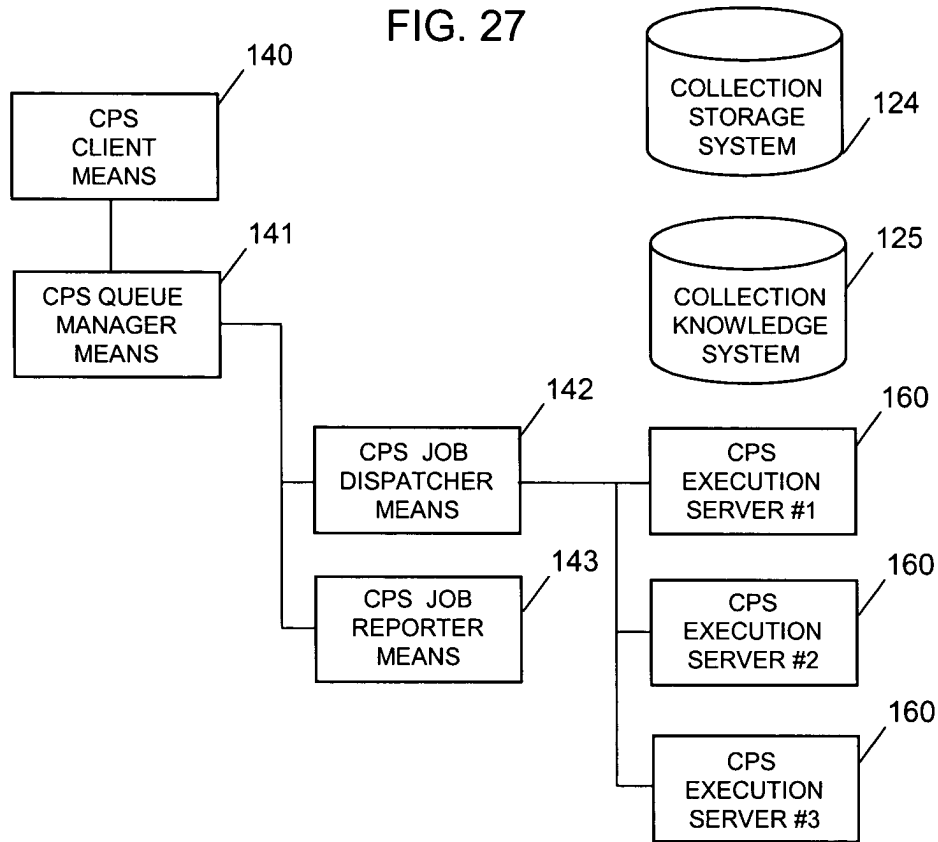


FIG. 28

- 1 # Simplified pseudocode algorithm for a collection processing system
- 2 CPS client receives job request, passes it to Queue Manager
- 3 Queue mgr calls external job expander to expand job request
- 4 - Q mgr calls job dispatcher, job dispatcher calls exe server
- 5 - Exe server calls external job expander program
- 6 - External job expander returns list of expanded job triplets
- 7 Queue mgr calls job dispatcher to expand and dispatch job triplets
- 8 Job dispatcher distributes second-level cmds among N exe servers
- 9 Job reporter means aggregates and reports job results
- 10 All modules may use CSS 124 and CKS 125, as required

FIG. 29

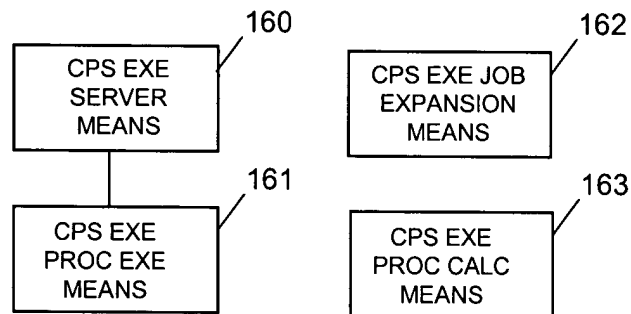


FIG. 30

```

1  # Simplified pseudocode algorithm for a CPS execution server
2  # that uses external program for job expansion
3  #
4  Receive 2nd level cmd from job dispatcher
5  If cmd is a job expansion cmd, call CPS exe job expansion means
6  - return expanded job triplets back to CPS Queue Manager
7  If cmd is a 2nd level prologue/main/epilogue cmd
8  - Expand 2nd level command into 3rd level command
5  - Call CPS Exe Process Execution Means to execute 3rd level cmd
6  Report job results back to CPS Job Dispatcher
  
```



16/20

FIG. 31

```
1  # syntax of a task definition file
2
3  taskpart <taskpart-name> [options] [attributes]
4
5  where
6  taskpart          - a label to make parsing easier
7  taskpart-name     - symbolic taskpart name
8
9  options           - task command options
10   sync            - wait until all platforms finish
11   one             - do command on only one platform
12   all            - do command on all platforms
13
14  attributes
15   fast            - run taskpart on a fast computer
16   native          - run taskpart on native platform
17                   (for running executables and tests)
18   make            - run taskpart using Make interpreter
19   perl            - run taskpart using Perl interpreter
20   <other>         - other user-defined attrs are possible
```

FIG. 32

```
1  name-symbolic-task.tbl:
2  # a table of symbolic task names
3
4  # name            definition file
5  rebuild          task-rebuild.def
6  export           task-export.def
7  regtest          task-reg-test.def
```



17/20

FIG. 33

```
1 task-rebuild.def:
2 # a task definition file for rebuilding a c program collection
3
4 # checkout coll from collection storage system using a fast cpu
5 # to avoid checkout collisions, only one platform does checkout
6 taskpart checkout fast one
7
8 # all platforms must wait until the checkout is done
9 taskpart sync sync
10
11 # generate a custom makefile containing third-level cmd seqs
12 taskpart gen-makefile
13
14 # compile and link the program locally
15 taskpart local make
16
17 # export built platform-independent files to a team shared tree
18 # to avoid collisions, one platform exports pi files first
19 taskpart export make one
20
21 # all platforms must wait until the export is done
22 taskpart sync sync
23
24 # now all other platforms can export
25 # no collisions now, since timestamps on pi files are up to date
26 taskpart export make
27
28 # epilogue commands
29 # whatever else users define
```



18/20

FIG. 34

```
1  cps-exe-vars.tbl:
2  # a table of variables defined by cps execution servers
3
4  REF          reference directory holding exe workspace
5  CAT          category part of collection reference
6  AUTH         authority part of collection reference
7  COLL         collection name part of collection reference
8  PLT          current platform of exe server
9  CMD-DIR      default directory to execute commands in
10 MAKE         name of make program on PLT
11 ...
```

FIG. 35

```
1  # example substitution string values for win2000 platform
2
3  REF          c:\cps\work
4  CAT          cf-colls
5  AUTH         mysite.com
6  COLL         c-program
7  PLT          win2000
8  MAKE         nmake
```

FIG. 36

```
1  # example substitution string values for linux2 platform
2
3  REF          /usr/local/cps/work
4  ...
5  PLT          linux2
6  MAKE         make
```





19/20

FIG. 37

```
1  name-task-part.tbl:
2  # a table of symbolic task PART names
3
4  # name          definition file
5  checkout        taskpart-checkout.def
6  gen-makefile    taskpart-gen-makefile.def
7  local           taskpart-local.def
8  export          taskpart-export.def
```

FIG. 38

```
1  taskpart-checkout.def:
2  # a task part definition file for checking out a collection
3  # syntax:
4  #  var <var-name> <var-value>
5  #  cmd <command-to-execute>
6
7  # all commands are run in directory $(CMD-DIR)
8  # variable substitutions are performed before execution
9
10 # make a directory in which to checkout the collection
11 cmd mkdir $(REF-CAT-DIR)
12
13 # set CMD-DIR to run all further commands in the new dir
14 var CMD-DIR $(REF-CAT-DIR)
15
16 # checkout a collection from a collection storage system (css)
17 cmd css checkout $(CAT):$(AUTH):$(COLL)
```





20/20

FIG. 39

```
1 taskpart-gen-makefile.def:
2 # a task part definition file for generating a makefile
3 # cmd <command-to-execute>
4 # var <var-name> <var-value>
5 # all commands are run in directory $(CMD-DIR)
6
7 # make a platform dir, execute commands in there
8 cmd mkdir $(PLT)
9 var CMD-DIR $(PLT)
10
11 # generate a makefile using the 'smnow' generator program
12 cmd smnow go
```

FIG. 40

```
1 taskpart-local.def:
2 # a task part definition file for building coll locally
3 # cmd</platform> <command-to-execute>
4 # var</platform> <var-name> <var-value>
5 # all commands are run in directory $(CMD-DIR)
6
7 # make a platform dir, execute commands in there
8 cmd mkdir $(PLT)
9 var CMD-DIR $(PLT)
10
11 # generate a makefile using the 'smnow' generator program
12 cmd clean make
13 cmd local make
```

